Swarm Intell (2008) 2: 143–165
DOI 10.1007/s11721-008-0012-6

SWARMORPH-script: a language for arbitrary morphology generation in self-assembling robots

Anders Lyhne Christensen · Rehan O'Grady · Marco Dorigo

Received: 21 December 2007 / Accepted: 11 June 2008 / Published online: 30 July 2008

Abstract In certain multi-robot systems, the physical limitations of the individual robots can be overcome using self-assembly—the autonomous creation of physical connections between individual robots to form a larger composite robotic entity. However, existing robotic systems capable of self-assembly have little or no control over the morphology of the self-assembled entities. This restricts the adaptability of such systems, since robots can carry out certain tasks more efficiently if their morphology is specialized to the task. In this paper, we extend the distributed mechanism presented in (Christensen et al. in IEEE Robot. Autom. Mag. 14(4):18–25, 2007) that allows autonomous mobile robots to self-assemble into specific morphologies. We present a simple language, SWARMORPH-script, that allows for concise descriptions of the rules that govern the distributed morphology growth process. Local visual communication allows physically connected robots to send and receive strings. A string can be a rule identifier that triggers execution of predefined logic for extending a morphology. Alternatively, whole scripts can be communicated and subsequently executed on the receiving robot. On real self-propelled robots capable of self-assembly, we demonstrate how specific morphologies can be constructed, how the size of a morphology can be regulated, and how multiple morphologies can be assembled. We also show how the transmission of entire scripts gives the robots the capacity to participate in the formation of morphologies of which they had no a priori knowledge.

Electronic supplementary material The online version of this article (http://dx.doi.org/10.1007/s11721-008-0012-6) contains supplementary material, which is available to authorized users.

A.L. Christensen (✉) · R. O'Grady · M. Dorigo
IRIDIA, CoDE, Université Libre de Bruxelles, 50 Av. Franklin Roosevelt CP 194/6, 1050 Brussels, Belgium
e-mail: alyhne@iridia.ulb.ac.be

R. O'Grady
e-mail: rogrady@ulb.ac.be

M. Dorigo
e-mail: mdorigo@ulb.ac.be

Keywords Morphogenesis · Self-replication · Self-assembly · Autonomous robots · Swarm robotics

1 Introduction

In multi-robot systems, individual robots can carry out different tasks in parallel. They can also cooperate when necessary. A particular way to cooperate is via self-assembly, a mechanism that allows teams of cooperating robots to overcome the physical limitations of the individual team members. Self-assembly involves the formation of physical connections between individual robots to create larger composite robotic entities. Figure 1 shows two examples of physically connected robotic entities carrying out tasks impossible for a single robot. These particular examples also highlight the importance of the morphology of the self-assembled robotic entities. The elongated structure (left) allows the robots to reach across the trough while the dense structure (right) provides stability on rough terrain. In this study, we propose a distributed algorithm for self-assembling multiple, autonomous robots into specific morphologies.

Self-assembly is challenging because it requires numerous autonomous robots with limited sensory capabilities to coordinate their actions. This becomes even more challenging when the control logic is distributed. Decentralized control is usually favored in multi-robot systems for the benefits it confers of scalability, robustness and flexibility (Bonabeau et al. 1999). However, distributed control renders the problem of coordination more difficult, as the individual agents usually have only a partial view of the system and have to act solely on the basis of local information.

In Christensen et al. (2007b), we presented a simple morphology control mechanism. Our longer term goal is to extend this system to the point where it will be able to form morphologies adaptively based on the demands of its task and environment. In this study, we take several important steps towards this goal. In particular, the contributions of this study are as follows. We show how arbitrary morphologies can be formed. We show how morphology size can be regulated. We demonstrate how multiple separate morphologies can be assembled. Finally, we show how robots with no a priori knowledge of a task can form morphologies based on instructions from robots already engaged in task-execution.

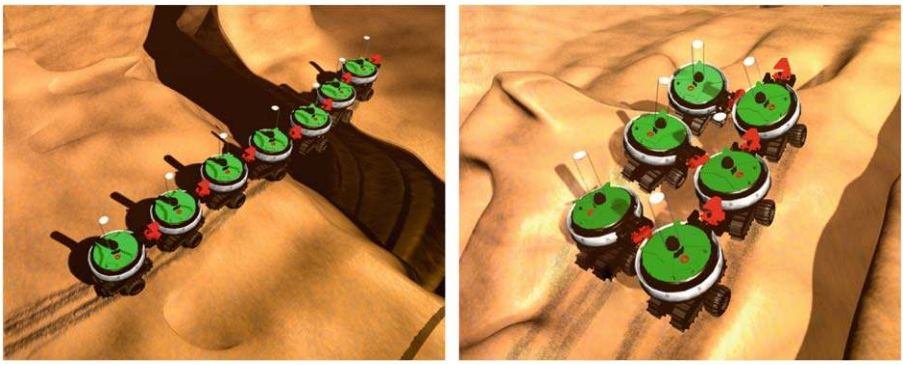


Fig. 1 Two examples of robotic entities self-assembled into morphologies appropriate for the task. *Left*: A connected robotic entity crosses a trough. A line formation is well-suited to this task, since it allows the entity to stretch further and minimizes the number of robots suspended over the trough. *Right*: A more dense structure provides greater stability for rough terrain navigation

The paper is organized as follows: In Sect. 2, we discuss the motivation for this work. In Sect. 3, we review some of the hardware systems capable of self-assembly and self-reconfiguration as well as proposed algorithms for morphology generation. In Sect. 4, we present the hardware platform used in this study, the SWARM-BOT platform, which consists of a number of autonomous mobile robots capable of self-assembly. In Sect. 5, we discuss a high-level language in which distributed morphology growth control logic can be described, and present the control primitives that the language relies on. In Sects. 6, 7, and 8, we demonstrate various morphologies constructed with real robots, highlighting different aspects of our system. In Sect. 9, we discuss the results obtained. Finally, in Sect. 10, we conclude and give directions for future research.

2 Motivation and methodology

In a previous work (Christensen et al. 2007b), we demonstrated the self-organized growth of specific morphologies. We subsequently identified several basic properties, not present in our previous system, that we believe are essential for a practically useful multi-robot morphology generation system.

Firstly, we would expect any practically useful morphology generation system to be able to form arbitrary morphologies. This means that the system must be able to form any morphology that the hardware of the system allows. Secondly, we would expect such a system to be able to control the size of the morphologies generated—at a certain point morphology growth must stop to allow the new self-assembled robotic entity to carry out its task. Thirdly, we would expect the generation of multiple morphologies to be possible—we would like separate self-assembled robotic entities to be able to carry out tasks in parallel. To cross an obstacle, for example, a specific morphology of three robots may be sufficient. If we had many hundreds of robots trying to cross the obstacle, it would normally make more sense to form many distinct morphologies of three robots rather than a single giant morphology.

Finally, we would like robots with no a priori knowledge of a task to be able to form morphologies based on instructions from robots already engaged in task-execution. This is important in adaptive task scenarios, where a finite number of pre-programmed morphologies may not be sufficient to respond to all of the environmental circumstances that may occur. In such cases, it may only be a subset of robots (perhaps only a single robot) that are in the right place at the right time to calculate the morphology that is needed for task completion. Robots that are not in this (possibly very small) subset will not be able to contribute to morphology generation unless the system has some mechanism to transmit knowledge of the desired morphology to robots that had no a priori knowledge of the task or the morphology.

In our previous work, connected robots followed local pattern extension rules to extend the local structure by inviting new connections in a particular direction. However, the absence of symbolic communication, as well as the homogeneity of the robotic controllers, meant that each newly connected robot had to follow the same pattern extension rules, with the result that only repeating structures were possible.

In this study, we retain the paradigm of morphology growth through local pattern extension. However, to enable the desirable system properties listed above, we make two key changes to the system. Firstly, we abstract our system into a set of control primitives. We use these control primitives to build a morphology creation language (SWARMORPH-script) that can be executed on real robots. The language allows for explicit high-level expression of distributed rules for morphology growth as opposed to implicit logic expressed in low-level source code. Secondly, we augment the system's communication capabilities to allow for the transmission of strings between physically connected robots.

With our new approach, arbitrary morphologies can be created by communicating information about the local state of the morphology (usually the communication of a single digit suffices) to each new robot that connects to the structure. The newly connected robot can extend the local structure appropriately based on this information, then pass on updated state information to the next robot to attach. A similar mechanism allows for morphology size regulation—a counter can be incremented and passed on by each attaching robot. Once the counter has reached a certain threshold, the next robot can stop extending the morphology.

Generation of multiple morphologies is enabled by adding 'disconnect' and 'retreat' control primitives to our morphology creation language. A robot that is part of a finished pattern can spark the creation of a new morphology by inviting a new connection to the already finished morphology. Once a robot has attached to this new connection, instructions to 'disconnect, retreat, and start a new morphology' are communicated to the newly attached robot.

Knowledge transmission about a priori unknown morphologies is achieved through the communication of sequences of morphology generation control instructions. Dedicated primitives in the SWARMORPH-script language allow for the communication of partial or whole morphology generation control programs.

3 Related work

In this section, we initially describe hardware systems with properties similar to those of the SWARM-BOT platform used in this study. We go on to mention some hardware systems in the related field of self-reconfigurable robotics. We then describe algorithms that have been proposed for controlling and assembling modular robots. We also mention some algorithms from the related field of formation control. Finally, we briefly discuss how our method differs from the algorithmic approaches in the existing literature.

The SWARM-BOT platform (Mondada et al. 2005) used in this study consists of a number of self-propelled robots that can self-assemble by forming physical connections with each other. The robots can either operate independently or can autonomously self-assemble and act as a larger robotic entity (the platform is described in detail in Sect. 4). Other systems in which the individual modules are mobile robots in their own right include Gunryu (Hirose et al. 1996), Super-Mechano Colony (SMC) (Damoto et al. 2001) and the Millibot Train system (Brown et al. 2002). Hirose et al. proposed Gunryu, an early concept for a system with independently mobile robots that could attach to each other. Although a prototype was built, the complete system was never demonstrated. The SMC is composed of one parent unit and several child units. The child units can attach to the parent unit and are responsible for the locomotion of the parent unit when assembled. In (Yamakita et al. 2003), the child units rearranged themselves to produce optimal tracking performance at different speeds. In the SMC Rover (Motomura et al. 2005), the child units are called Uni-Rovers. Each Uni-Rover is composed of a wheel and a single manipulator. The Uni-Rovers can attach to the parent unit at one of six locations. When six Uni-Rovers connect to the parent unit, the artifact effectively becomes a wheeled rover. In the Millibot Train system (Brown et al. 2002), multiple mobile robots are physically connected in a train formation, so that they can cross obstacles that a single robot would not be able to cross. So far, only a teleoperated train composed of seven prototype modules has been demonstrated (Brown et al. 2002).

In addition to these systems, there is a large body of research in the related field of self-reconfigurable modular robotic systems. These systems explore morphological flexibility through the use of connected modular components. The individual modules vary in autonomy and independence of control from system to system. However, in contrast with

multi-robot systems, the individual modules are not capable of carrying out meaningful tasks independently. Self-reconfigurable modular robotic systems include Fukuda et al.'s CEBOT (Fukuda et al. 1991; Kawauchi et al. 1993), Yim et al.'s PolyBot (Yim et al. 2000, 2003), Castano et al.'s CONRO (Castano et al. 2000), Murata et al.'s M-TRAN (Murata et al. 2002), and Shen et al.'s SuperBot (Shen et al. 2006; Salemi et al. 2006). For detailed overviews of self-reconfigurable systems and self-assembling systems see Yim et al. (2007) and Groß and Dorigo (2008).

Various approaches to controlling and assembling modular robots have been proposed. Challenges include forming an appropriate morphology and getting numerous autonomous units to operate efficiently as a single entity. Rus and Vona (1999, 2001) have proposed a centralized control algorithm to allow a robotic system composed of *Crystalline Atom* units to reconfigure its shape. Although a physical prototype of a *Crystalline Atom* unit has been built, the results with the proposed control algorithm were obtained analytically and in simulation.

In White et al. (2005), an approach is proposed where individual cubic units have unique IDs and a predefined location in the structure. A scripting language is used to specify where each unit should connect.

Jones and Matarić (2003) have developed a *Transition Rule Set* compiler that takes as input the desired morphology and outputs a set of rules. Agents with limited and local sensing can follow these rules and assemble into the desired morphology. The approach proved scalable and capable of producing a large class of structures in a simulated 2D lattice world populated by simulated unit square agents.

Butler et al. (2004) have studied generic decentralized algorithms for lattice-based self-reconfigurable robots. The algorithms are inspired by cellular automata and control is based on geometric rules.

Klavins et al. (2006) have tackled the problem of defining a class of graph grammars that can be used to model and direct distributed robotic self-assembly. The authors show how a grammar that generates a desired, pre-specified target morphology can be synthesized.

Shen et al. (2004) have demonstrated a bio-inspired control method based on virtual hormone for controlling swarms of robots. The virtual hormone can be propagated through the swarm causing the robots to generate patterns and/or reconfigure. The authors show results from experiments in simulation and discuss how virtual hormone could be propagated in real-world scenarios either through radio or through infrared communication.

Again in simulation, Støy and Nagpal (2004, 2006) have demonstrated algorithms for self-reconfiguration and directed growth of cubic units based on gradients and cellular automata. Bojinov et al. (2000) have shown how a simulated modular robot (Proteo) can self-reconfigure into useful and emergent morphologies when the individual modules use local sensing and local control rules. Some of the practical and theoretical implications of adapting the shape of a self-reconfigurable robot to its environment have been studied by Yu et al. (2007) and Yu and Nagpal (2008).

The morphological structures we generate in this study are also reminiscent of the arrangements of robots studied in the related research field of multi-robot formation control (Balch and Arkin 1998; Das et al. 2002; Lawton et al. 2003; Lewis and Tan 1997). Proposed approaches to formation control include the use of virtual structures, leader-follower schemes, and decentralized, behavior-based methods. In formation control research, groups of robots steer themselves into one or more pre-specified formations. Mechanisms to maintain these formations while the group is in motion are also studied. By contrast, in morphogenesis research the shape of the composite entity is maintained automatically as a result of the physical connections between the robots.

In our decentralized system, the robots have no fixed IDs from the onset of an experiment. Instead, the robots specialize when they connect to a morphology. In some systems, the necessary specialization is achieved (a priori) by using heterogeneous robotic controllers. For example, in Shen et al.'s approach, different robots respond differently to the virtual hormone that is propagated through the system. In our approach, information about locally required expansion propagates through the system to the edges of the morphology, and our homogeneous robots specialize as they attach to the morphology and receive information about the local morphology extension required. Jones and Matarić (2003) proposed a compiler that maps a morphology to a set of rules. Butler et al. (2004) studied generic decentralized control algorithms for self-reconfigurable robots. Both of these studies dealt with lattice-based systems. In our system, each unit can connect to at most one other unit resulting in tree-like structures. We have written the rules by hand for the morphologies in this study. A compiler that automatically generates the rules for a desired morphology would, however, be trivial to implement for the type of structures that can be assembled with our robots.

4 Hardware platform

For our experiments, we use the innovative SWARM-BOT robotic platform (Mondada et al. 2004) created by Francesco Mondada's group at the Laboratoire de Systèmes Robotiques of the École Polytechnique Fédérale de Lausanne. The platform consists of a number of mobile autonomous robots called *s-bots* (see Fig. 2) that are capable of forming physical connections with each other. Thanks to its traction system that combines tracks and wheels,

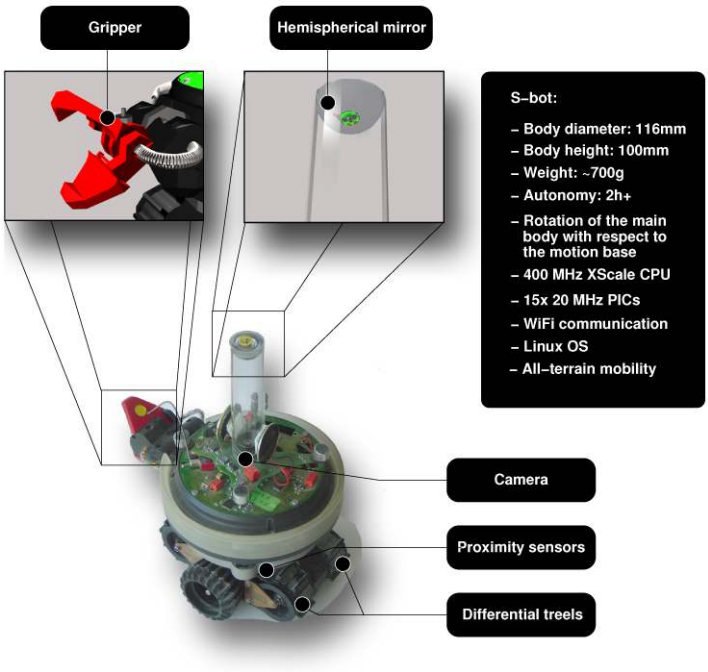


Fig. 2 *S-bot*: An autonomous, mobile robot capable of forming physical connections with other *s-bots*

the *s-bot* has good mobility on moderately uneven terrain while still retaining the ability to rotate on the spot efficiently.

The SWARM-BOT platform has been used for several studies, mainly in swarm intelligence and collective robotics (see, for instance, Dorigo et al. 2004, 2006; Trianni and Dorigo 2006). Overcoming steep hills and transport of heavy objects are notable examples of tasks which a single robot could not complete individually, but which have been solved successfully by teams of collaborating robots (Groß et al. 2006; O'Grady et al. 2005; Nouyan et al. 2006).

Each *s-bot* is equipped with an XScale CPU running at 400 MHz, a number of sensors including an omni-directional camera, infrared ground sensors and proximity sensors. Physical connections between *s-bots* are established by a gripper-based connection mechanism. Each *s-bot* is surrounded by a transparent ring that can be grasped by other *s-bots*. *S-bots* can advertise their location by means of eight sets of RGB-colored LEDs distributed around the inside of their transparent ring.

4.1 The *s-bot* camera and image processing

The *s-bot* has omni-directional vision, achieved using a camera that points upwards at a hemispherical mirror. A transparent perspex tube holds the mirror in place. The camera captures images of the robot's surroundings reflected in the hemispherical mirror. When the *s-bots* operate on flat terrain, the distance in pixels from the center of an image to a perceived object can be used to estimate the physical distance between the robot and the object (see Fig. 3).

The camera sensor records 640 × 480 color images. The *s-bots* have sufficient on-board processing power to scan entire images and identify objects based on color. Images are divided into a grid of multi-pixel blocks and the image processor outputs the prevalent color in each block (or indicates the absence of any relevant color). We have configured the image processor to detect the location of the colored LEDs of the *s-bots* and discard any other information. Depending on light conditions, an *s-bot* can detect LEDs on neighboring *s-bots* up to 50 cm away.

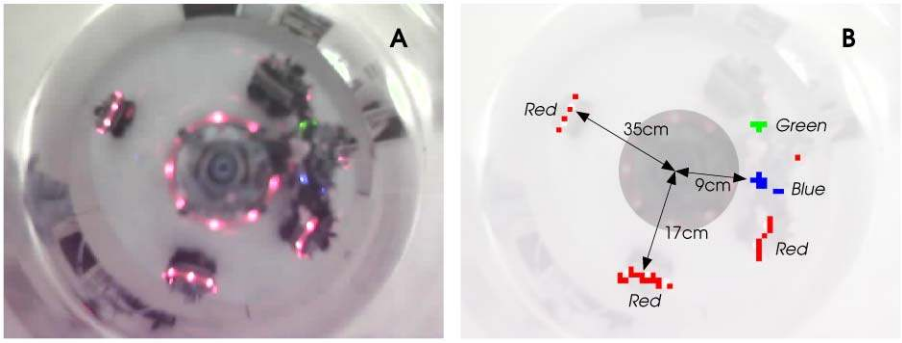


Fig. 3 **A** An example of an image captured by a robot's omni-directional camera. **B** The same image after color segmentation with indications of the distance estimates from the robot that captured the image to some of the LEDs detected

5 SWARMORPH-script

In this study, our goal is to assemble morphologies using distributed control logic on real robots. This means that each robot operates completely autonomously. Due to their limited sensory range, the robots attaching to the morphology do not have any knowledge of the current state of the global morphology, that is, they do not know how many robots are already connected in the morphology, nor do they know the orientation of the morphology with respect to their own position. Robots that are not part of the morphology cannot, therefore, deduce where they should attach. Instead, robots that are already part of the morphology use their colored LEDs to indicate where and with which orientation new robots should connect in order to extend the morphology correctly.

When a new robot connects to the morphology, it initiates communication with the robot to which it connected. Through this communication, the newly connected robot receives instructions about how to extend the local structure. Following these instructions, the newly connected robot in turn attracts other robots by lighting up its own colored LEDs. When a subsequent new robot attaches, it once again initiates communication, and is told in turn how to extend the structure. As this process repeats itself, the morphology grows accordingly.

We formulated a language called SWARMORPH-script which we use to describe distributed morphology control at a high level. SWARMORPH-script is based on a set of control primitives—abstractions of the basic behaviors needed to assemble morphologies. Below, we provide a description of the commands available in the language and the corresponding behaviors.

5.1 Directional self-assembly—`OpenConnSlot()`, `SearchForConnSlot()`

To extend a morphology locally in a particular direction, our system relies on directional self-assembly. This involves use of the *connection slot* mechanism. To grow the morphology, an *s-bot* that is already connected to the morphology 'opens' a connection slot by illuminating its LEDs in a particular color configuration. We refer to such an *s-bot* as an *extending s-bot*. This color configuration indicates a grip point on the extending *s-bot*'s body where another non-attached *s-bot* should grip and a corresponding orientation which the gripping *s-bot* should assume. We refer to an *s-bot* that tries to form an attachment to an extending *s-bot* as a *connecting s-bot*. The connection slot mechanism is illustrated in Fig. 4.

The command `OpenConnSlot` opens a connection slot. The slot that should be opened is given as a parameter to the command. This parameter can take the following values: `front-left`, `left`, `back-left`, `back`, `back-right`, `right`, `front-right`. Once an extending *s-bot* has opened a connection slot, it waits until another robot has connected to the connection slot. When a robot executes `SearchForConnSlot`, it tries to navigate to the nearest open connection slot and attempts to connect.

Details about the directional self-assembly mechanism, including a description of the logic for navigating to and approaching a connection slot as well as results about the precision of the mechanism on real *s-bots* are provided in (Christensen et al.2007a, 2007b).

5.2 Search—`RandomWalk()`

When a robot performs a random walk, it moves in straight lines and performs simple obstacle avoidance based on readings from its proximity sensors. The direction and length of the straight lines are chosen randomly. A robot continues to perform the random walk until it detects a hole in the ground (using its infrared ground sensors) or until it sees a connection slot (using its camera). Once either of these events has occurred, the robot executes the subsequent commands in its control script.

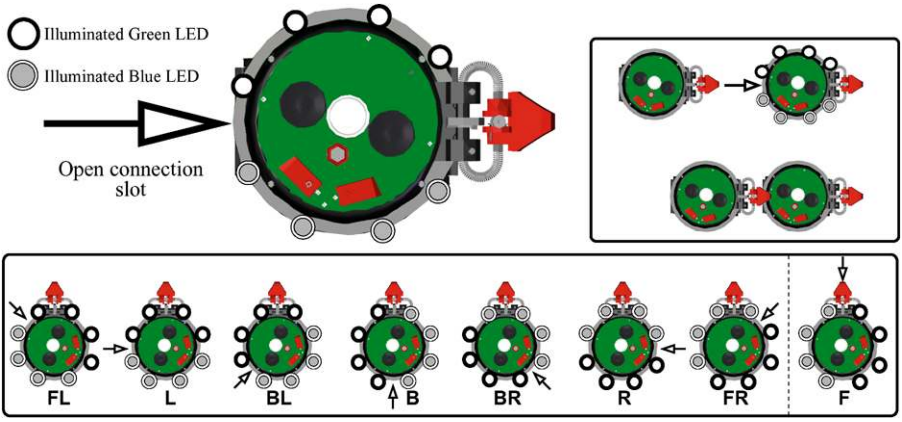


Fig. 4 *Above left*: An *s-bot* with an open connection slot to its rear. To open the connection slot the *s-bot* illuminates its left green LEDs and its right blue LEDs. *Above right*: An *s-bot* connecting to the `back` connection slot. *Below*: The 8 possible connection slots that an extending *s-bot* can open. Only seven connection slots can be used for extending the morphology. The eighth connection slot (F) indicates a grip point that is occupied by the gripper of the extending *s-bot*. This connection slot is used instead by a connecting robot to signal the extending *s-bot* that the connection slot has been filled. An example of an *s-bot* connecting to a connection slot and signaling to the extending *s-bot* is shown in Video 1—*Communication* in the online supplementary material

5.3 Local communication—`SendRuleID(rule-id)`, `ReceiveRuleID()`

Our morphology generation approach relies on local communication between physically connected robots. The commands `SendRuleID` and `ReceiveRuleID`, respectively, send and receive a number identifying a predefined rule. When a robot connects to a connection slot, the robot displaying the connection slot typically communicates to the newly attached robot how to extend the local structure. The most efficient way of doing this (that requires the least communication) is for the sending robot to transmit a single number—a rule ID. The receiving robot receives the number, then it follows the control logic (a set of script commands) associated with that rule. On the receiving robot, the association between the rule number received and the corresponding sequences of SWARMORPH-script instructions is achieved using the branching construct (explained below).

The *s-bot* has no hardware dedicated to local point-to-point communication and we have therefore implemented a simple protocol based on visual communication: We use the three colors red, green, and blue. Each time a bit is transmitted, the sending robot changes the illumination of its LEDs. The color green represents a '0' bit, blue represents a '1', and red represents a repeat bit. We rely on acknowledgment to distinguish adjacent bits. The receiver acknowledges receipt of each bit by lighting up its LEDs to match the color of the sender's LEDs. Once the receipt of a bit has been acknowledged, the sender transmits the next bit. This acknowledgment mechanism necessitates our use of the dedicated color for a repeat bit.

When transmitting a substring of two or more bits of the same value, every other bit will be represented by the color red (starting from the second bit). As an example, assume that the sender transmits the binary string '00': It first lights up its green LEDs to send the first '0'. When the receiver has acknowledged receiving '0' by lighting up its green LEDs, the sender lights up its red LEDs to indicate that the new bit has the same value as the previous one. In this way, every new bit causes a change in color that can be acknowledged by the

receiver. If the sender transmits the binary string '111', it first lights up its blue LEDs to communicate the first '1', then its red LEDs to communicate the second '1', and finally blue again to transmit the third '1'.

To indicate the end-of-transmission, the sender turns off its LEDs. When the receiving robot detects end-of-transmission, it performs a simple validity check by counting the number of bits received. We transmit each number as a sequence of 5 bits. This allows us to perform a simple error check—the receiver checks at the end of a transmission that the number of bits received is a multiple of 5. The receiver illuminates its green LEDs if the length check indicated that the string was successfully received. Otherwise it illuminates its blue LEDs to request retransmission.

We tested the communication mechanism on real robots (see Video 1—Communication in the online supplementary material), and it proved reliable but relatively slow: We transmitted strings of 100 bits 10 times. In all 10 trials, the string was successfully communicated and on average the transmission took 121 seconds resulting in an average bandwidth of 0.8 bits/second. On other robot platforms more efficient means for local and situated communication (Støy 2001) might be available, such as infrared or communication through the mechanism that connects different modules (see, for instance, Murata et al. 2002). For this study, however, the visual communication protocol suffices to demonstrate our approach on real robots.

5.4 Communication of whole scripts—`SendScript(command-list | 'self')`, `ReceiveScript()`, `ExecuteReceivedScript()`

The communication mechanism described above can communicate arbitrary strings. By creating an appropriate mapping, we use this mechanism to communicate SMARMORPH-scripts. To send a script, each SWARMORPH-script command is compiled to a 5-bit string literal. The robot that receives a script maps these 5-bit string literals back to SWARMORPH-script commands, and then executes the script thus received. The `SendScript` command takes a list of SWARMORPH-script commands as a parameter. Alternatively, a special parameter, `'self'`, can be provided to the `SendScript` command which tells the sending robot to transmit a copy of the whole script it is currently executing.

5.5 Disconnect and retreat—`Disconnect()`, `Retreat(time-out)`

The commands `Disconnect` and `Retreat`, respectively, allow a robot connected to a morphology to disconnect (by opening its gripper) and to move backwards for a period of time. In this way, when a morphology is complete, a robot in the morphology can open a temporary connection slot and instruct the robot that connects to the slots to disconnect and start a new morphology.

5.6 Miscellaneous—branching (`if, then, end`), `StopExecution()`

We have added a simple branching construct:

if *[condition]* **then**
    [sub-script]
end

The [sub-script] is executed if and only if *[condition]* is true. The condition is typically whether or not a feature, such as a hole, has been detected in the environment.

The command `StopExecution` causes the execution of the script to stop.

5.7 Example

An example script for a robot that starts a morphology is shown in Script 1. The extending *s-bot* first opens a connection slot to its rear. When another robot connects to the connection slot, the robot executing the script sends a '1' telling the newly connected robot to execute rule number 1. An example of a corresponding script for a connecting robot is shown in Script 2.

A robot executing the script shown in Script 2 first searches for a connection slot and attempts to connect. Once a successful connection has been formed, the robot expects to receive the ID of a rule from the robot to which it connected. If it receives a '1', it opens a connection slot to its rear and waits for another robot to connect. If it receives a '2', it calls the `StopExecution` command and takes no further action.

An example morphology grown with three robots is shown in Fig. 5. One robot starts the morphology by executing Script 1 and two robots connect to the morphology by executing Script 2. The first robot opens a connection slot to its rear (Fig. 5A) and sends rule ID 1 to the first robot that connects (Fig. 5C). The first robot to connect in turn opens a connection slot to its rear, and when the connection slot is filled (Fig. 5D), it sends rule ID 2 to the second (and last) robot that connects to the morphology (Fig. 5E). The last robot does not

Script 1: Simple example of a script for a robot that starts a new morphology.

```
OpenConnSlot(back);        // Open a connection slot in the back and
                           // wait for an s-bot to connect.
SendRuleID(1);             // Send rule ID "1" to tell the connected
                           // s-bot to execute rule number 1.
StopExecution();           // Stop processing
```

Script 2: Simple example of a script for a robot that connects to an existing morphology and extends it.

```
SearchForConnSlot();       // Find and connect to a robot with an open
                           // connection slot.
ReceiveRuleID();           // Initiate communication and receive a
                           // rule ID.
if receivedruleid = 1 then
    OpenConnSlot(back);        // Open a connection slot in the back and
                               // wait for an s-bot to connect.
    SendRuleID(2);             // Send rule ID 2 to tell the connected
                               // s-bot to execute rule 2.
    StopExecution();           // Stop processing.
end
if receivedruleid = 2 then
    StopExecution();           // Stop processing.
end
```

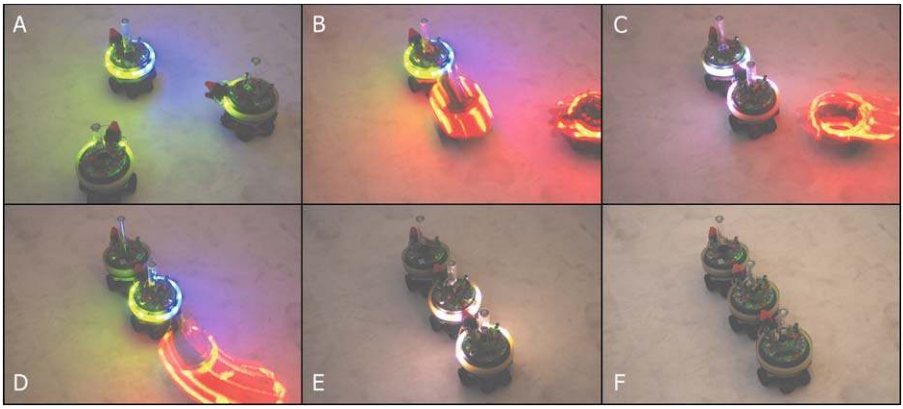


Fig. 5 Result when one robot executes Script 1 and two robots execute Script 2. See text for details

open any connection slots and the morphology is complete (Fig. 5F). The result is a line morphology.

6 Morphogenesis using local communication and predefined rules

In this section, and in Sects. 7 and 8, we present our results: specific morphologies formed with real robots. Every photograph of a morphology presented in these sections is the final outcome of an actual experimental trial. Videos of each trial showing the autonomous incremental growth of the morphologies in each of the photographs are provided in the online supplementary material.

We demonstrate morphogenesis using local communication and predefined rules with three morphologies—the *horseshoe*, the *shovel*, and the *square*. A 'rule' is a preprogrammed sequence of instructions that tells an *s-bot* how to extend the local structure.

For each morphology we used homogeneous independent controllers on 6 to 11 real *s-bots*. The morphology is seeded when one of the *s-bots* detects an obstacle. This *s-bot* becomes the seed for the morphology and opens the first connection slot. The other *s-bots* see the open connection slot, and attempt to connect to the morphology. Once an *s-bot* has seen a connection slot, it can no longer become a seed. The 'obstacle' in our case is a dark patch on the floor of the arena representing a hole.[1] At the start of each experiment, we place the *s-bots* pointing towards a 10 cm $\times$ 10 cm dark patch on the floor in the center of the arena (see Fig. 6).

The basic structure of any SWARMORPH-script that uses this obstacle-based seeding is shown in Script 3. The function `RandomWalk` performs a random walk until either a hole (the dark patch) or a connection slot is seen. If a hole is detected, the logic in the **if** scope for starting the morphology is executed. Otherwise, if a connection slot is seen, the robot attempts to connect to the existing morphology and waits to receive a rule ID number. This

[1] In this study our focus is on morphology control rather than navigation. We therefore use a dark patch in place of the hole to prevent robot casualties. In the future we will implement more sophisticated logic to ensure safe navigation in environments containing various types of obstacles, such as holes. To the *s-bot's* ground sensors, a dark patch on the floor is indistinguishable from a hole.

Fig. 6 Starting configuration for experiments using obstacle-based seeding

Script 3: Basic structure of a script using local communication and predefined rules.

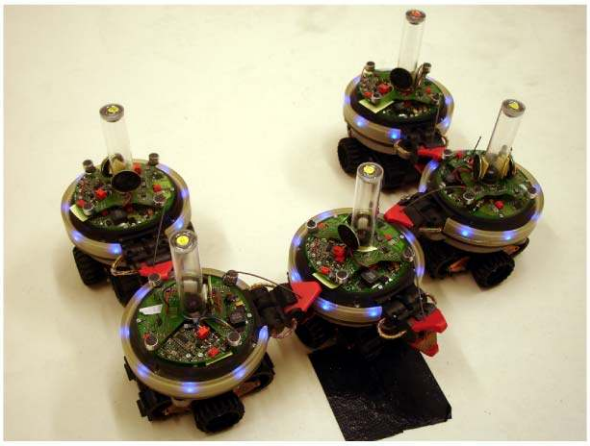
```
// Random walk until a hole or a connection slot is detected
RandomWalk();

// If a hole is detected, a new morphology is started
if hole-detected then
    ...
    Start the morphology.
    ...
    StopExecution();
end

// If a connection slot is detected, an attempt is
// made to connect to the morphology.
if conn-slot-detected then
    SearchForConnSlot();
    ReceiveRuleID();
    ...
    Execute morphology-specific rules.
    ...
end
```

number will map to a particular sequence of pre-defined commands that will tell the robot how to extend the structure locally.

The *horseshoe, shovel*, and *square* morphologies and the corresponding SWARMORPH-scripts we used to generate them are shown in Figs. 7, 8, and 9. In one experiment, we manually reset a robot that had suffered a low level software failure. Faulty proximity sensors on a few of the robots resulted in occasional robot collisions—in some cases this resulted in one of the robots toppling over. When this happened we manually righted the toppled robot. On average, a robot toppled over once for every 661 seconds of experimentation time (this figure applies to all of the experiments presented in this paper, not just the experiments in this section).



```
RandomWalk();
if hole-detected then
    OpenConnSlot(left);
    SendRuleID(1);
    OpenConnSlot(right);
    SendRuleID(2);
    StopExecution();
end
if conn-slot-detected then
    SearchForConnSlot();
    ReceiveRuleID();
    if receivedruleid = 0 then
        StopExecution();
    end
    if receivedruleid = 1 then
        OpenConnSlot(right);
        SendRuleID(0);
    end
    if receivedruleid = 2 then
        OpenConnSlot(left);
        SendRuleID(0);
    end
end
```

Fig. 7 *Horseshoe* script and result. Video 2—*Rule-based Morphologies* in the online supplementary material shows the experiment which produced this result

In each of these experiments, whenever a new robot connects to the morphology, it receives the ID of a rule that indicates how the local structure should be extended.

In the *horseshoe* (Fig. 7) and the *shovel* (Fig. 8), three different extension rules are used: one for the robot to the left of the seed (rule ID 1), one for the robot to the right of the seed (rule ID 2), and one further rule to stop the morphology growth locally. Both morphologies require three extension rules. In our previous approach presented in Christensen et al. (2007b), a connecting robot had to decide how to extend the local structure based on whether the robot to which it connected opened another connection slot or not. This allowed for only two different extension rules and neither the *horseshoe* nor the *shovel* could thus have been formed using this approach. The approach presented in the current study gives the robots the capacity to communicate explicit rule IDs and there is therefore no (practical) limit to the number of extension rules that we can use. Arbitrary morphologies can thus be formed.

The *square* morphology is generated from three extension rules like the *horseshoe* and the *shovel* described above. However, in the *square* (Fig. 9), we take advantage of the symmetry in the structure, and rule ID 2 is sent to both the third and fourth robot that attach to the seed. We could, in fact, have built a square using two extension rules only, by having the seed send rule ID 2 to all the robots that connect to it. The outline of the resulting morphology would be the same, but the topology of connections between the robots would be different.

7 Multiple morphologies

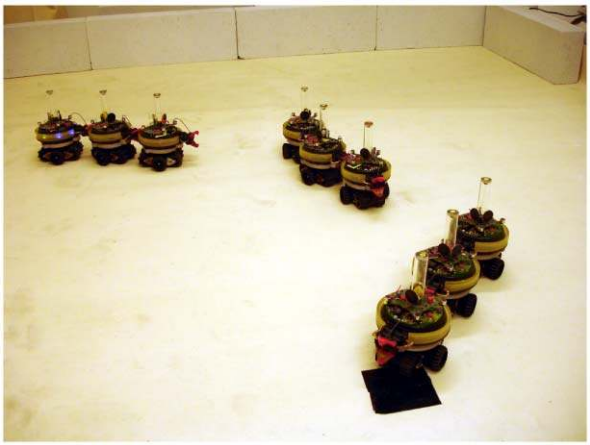
In practical task-execution scenarios with large numbers of robots, the formation of multiple morphologies is likely to be a requirement. As we increase the number of self-assembling robots in a robotic swarm, the need for the self-assembled robots to have an appropriate morphology does not change. However, rather than generating a single giant morphology, it

```
RandomWalk();
if hole-detected then
    OpenConnSlot(back);
    SendRuleID(0);
    OpenConnSlot(left);
    SendRuleID(1);
    OpenConnSlot(right);
    SendRuleID(2);
    StopExecution();
end
if conn-slot-detected then
    SearchForConnSlot();
    ReceiveRuleID();
    if receivedruleid = 0 then
        StopExecution();
    end
    if receivedruleid = 1 then
        OpenConnSlot(right);
        SendRuleID(0);
        OpenConnSlot(back-left);
        SendRuleID(0);
    end
    if receivedruleid = 2 then
        OpenConnSlot(left);
        SendRuleID(0);
        OpenConnSlot(back-right);
        SendRuleID(0);
    end
end
```

Fig. 8 *Shovel* script and result. Video 2—*Rule-based Morphologies* in the online supplementary material shows the experiment which produced this result

```
RandomWalk();
if hole-detected then
    OpenConnSlot(front-left);
    SendRuleID(1);
    OpenConnSlot(back-left);
    SendRuleID(0);
    OpenConnSlot(back-right);
    SendRuleID(2);
    OpenConnSlot(front-right);
    SendRuleID(2);
    StopExecution();
end
if conn-slot-detected then
    SearchForConnSlot();
    ReceiveRuleID();
    if receivedruleid = 0 then
        StopExecution();
    end
    if receivedruleid = 1 then
        OpenConnSlot(left);
        SendRuleID(0);
        OpenConnSlot(right);
        SendRuleID(0);
    end
    if receivedruleid = 2 then
        OpenConnSlot(left);
        SendRuleID(0);
    end
end
```

Fig. 9 *Square* script and result. Video 2—*Rule-based Morphologies* in the online supplementary material shows the experiment which produced this result



```
RandomWalk();
if hole-detected then
    OpenConnSlot(back);
    SendRuleID(1);
    StopExecution();
end
if conn-slot-detected then
    SearchForConnSlot();
    ReceiveRuleID();
    if receivedruleid = 1 then
        OpenConnSlot(back);
        SendRuleID(2);
    end
    if receivedruleid = 2 then
        OpenConnSlot(right);
        SendRuleID(3);
    end
    if receivedruleid = 3 then
        Disconnect;
        Retreat(5 s);
        OpenConnSlot(back);
        SendRuleID(1);
    end
end
```

Fig. 10 *Lines-of-three* script and result. Video 3—*Multiple Morphologies* in the online supplementary material shows the experiment which produced this result

is more reasonable to assume that many smaller morphologies will be appropriate. This will allow parallel execution and teamwork at the level of self-assembled robotic entities.

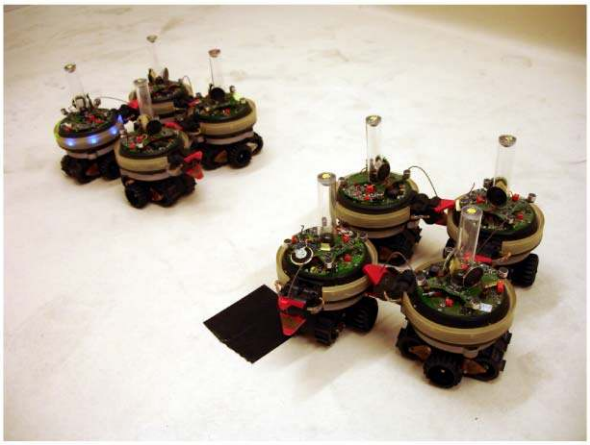
The generation of multiple morphologies in our system is enabled through a morphology replication mechanism. Once a morphology has finished forming, a robot that is part of a finished pattern can spark the creation of new morphology by opening a new temporary connection slot. This connection slot is temporary, because the robot that attaches to it does not stay attached. Instead, it receives a rule ID that it maps to instructions to retreat and then start a new identical morphology from scratch.

In this section, we demonstrate the generation of multiple morphologies by generating multiple copies of two morphologies—the *lines-of-three* morphology (Fig. 10) and the *mini-squares* morphology (Fig. 11). In both cases, the experimental setup is the same as in the previous section, including the use of obstacle-based seeding.

In the *lines-of-three* morphology, a linear morphology consisting of three *s-bots* is initially constructed. The third and final robot in the morphology opens a temporary connection slot. When the fourth robot connects to the morphology, it receives rule ID '3' that causes it to disconnect, reverse for 5 s and then start a new *lines-of-three* morphology.

The *mini-squares* morphology utilizes the same principle. Each mini-square consists of four robots. The fifth robot to connect disconnects, retreats, and starts the morphology growth process again. For both the line morphology and the mini-square morphology, the formation process continues until no more robots are available.

We chose to generate several copies of simple morphologies in order to demonstrate the generation of multiple morphologies. In principle, there is nothing preventing the generation of multiple complex and/or different morphologies. If we had sufficient robots available, we could have made multiple *horseshoes*, *shovels*, and *squares* in the same way.



```
RandomWalk();
if hole-detected then
    OpenConnSlot(left);
    SendRuleID(1);
    StopExecution();
end
if conn-slot-detected then
    SearchForConnSlot();
    ReceiveRuleID();
    if receivedruleid = 1 then
        OpenConnSlot(right);
        SendRuleID(2);
    end
    if receivedruleid = 2 then
        OpenConnSlot(right);
        SendRuleID(3);
    end
    if receivedruleid = 3 then
        OpenConnSlot(back);
        SendRuleID(4);
    end
    if receivedruleid = 4 then
        Disconnect;
        Retreat(5 s);
        OpenConnSlot(left);
        SendRuleID(1);
    end
end
```

Fig. 11 *Mini-squares* script and result. Video 3—*Multiple Morphologies* in the online supplementary material shows the experiment which produced this result

8 Towards adaptive morphogenesis: control logic transmission

In this section, we show how robots with no a priori knowledge of the morphology being formed can nonetheless take part in morphology growth. To do this, we rely on the transmission of control logic, that is, on the communication of whole or partial SWARMORPH-script programs between connected robots. In the previous two sections, the communication was limited to the transmission of a single number—a rule ID that mapped to segments of preprogrammed control logic on the recipient robot. This approach is efficient in terms of restricting communication to a minimum. However, the set of morphologies that can be generated is limited by the set of rules given to the robots at the start of an experiment.

In some adaptive task-execution scenarios, it is infeasible to store a finite set of pre-programmed morphologies to match every possible environmental contingency. Imagine, for example, a scenario in which one robot encounters a previously unseen obstacle. By analyzing the obstacle, the robot calculates the appropriate morphology to overcome the obstacle. This morphology is a function of the obstacle encountered—the morphology cannot be pre-programmed as there are far too many obstacle shapes that may be encountered.

In scenarios in which pre-programmed morphologies are not feasible, rule IDs are insufficient as a means to communicate which morphology should be formed. The transmission of whole or partial SWARMORPH-script control programs solves this problem. In the example above, the robot that encounters the obstacle above can go and recruit other robots (that have no knowledge of the obstacle and therefore no knowledge of the appropriate morphology) by sending the other robots a script with instructions on how to construct the relevant morphology.

Script 4: Script executed by the robots without any preprogrammed morphology expansion rules.

```
SearchForConnSlot();
ReceiveScript();
ExecuteReceivedScript();
```

We demonstrate the communication of morphology control logic with three different morphologies—the *communicated-line-replicator*, the *communicated-triangle-replicator*, and the *communicated-step-replicator*. Our experimental setup mimics the scenario described above: we program a single 'master' robot with morphology control logic to create a morphology and to communicate the same control to other robots. All other robots are 'slave' robots that start without any a priori morphology creation logic, except the basic logic required to attach to a connection slot and execute any morphology control logic received through communication. The SWARMORPH-script that we execute on these 'slave' robots is shown in Script 4.

To demonstrate the communication of both partial control programs and whole control programs, we use replicating morphologies. Partial control programs to form segments of the morphology are communicated as the morphologies are developing. The whole control program (to generate the entire morphology and subsequently to repeat the replication process) is communicated when morphology replication occurs. The three *replicator* scripts (as executed on the seed robot) and the result when running them on real *s-bots* are shown in Figs. 12, 13, and 14, respectively.

In the *communicated-triangle-replicator* (Fig. 12), the seed robot executes the `Disconnect` and `Retreat` instructions before opening a connection slot. This is true for all of the morphologies in this section, and is done to maintain homogeneity of control, as these are the first instructions that a 'slave' robot will need to execute when it receives instructions to replicate the morphology. The seed robot then opens a connection slot to its rear left and sends a `StopExecution` command to the connecting robot. Thus, the first connected robot takes no further actions. The seed then opens a second connection slot to its rear right and sends a `StopExecution` command to the robot that connects to the slot. When three robots are assembled in this way, their morphology resembles a triangle (see Fig. 12). Once the three robots are assembled the seed opens a temporary connection slot to its front left and sends a copy of its own script (`SendScript(self)`) to the robot that connects. This connecting robot disconnects, retreats and restarts the process.

In the *communicated-step-replicator* (Fig. 13), the seed robot first opens a connection slot to its left and sends a SWARMORPH-script to the connecting robot that instructs it to open a connection slot to its left and send a `StopExecution` command to the robot that connects. The seed robot then opens a connect slot to its right and sends a copy of its own script to the robot that connects.

The *communicated-line-replicator* (Fig. 14), uses a similar principle—a SWARMORPH-script is sent to the first robot that connects which in turn opens a new connection slot and sends a `StopExecution` command to the robot that connects. The seed robot then opens another connect slot and sends a copy of its own script to the robot that connects, and a new line morphology is started. The result of the *communicated-line-replicator* is similar to the *lines-of-three* rule-based script (see Figs. 14 and 10), but whereas all the robots in the *lines-of-three* experiment had the complete set of rules for the morphology, only a single robot had rules for generating line morphologies at the start of the *communicated-line-replicator* experiment.

```
Disconnect();
Retreat(10 s);
OpenConnSlot(back-left);
SendScript(
   StopExecution();
);
OpenConnSlot(back-right);
SendScript(
   StopExecution();
);
OpenConnSlot(front-left);
SendScript(self);

StopExecution();
```

Fig. 12 *Communicated-triangle-replicator* script and result. Video 4—*Towards Adaptive Morphogenesis* in the online supplementary material shows the experiment which produced this result

```
Disconnect();
Retreat(10 s);
OpenConnSlot(right);
SendScript(
   OpenConnSlot(left);
   SendScript(
        StopExecution();
   );
   StopExecution();
);
OpenConnSlot(left);
SendScript(self);

StopExecution();
```

Fig. 13 *Communicated-step-replicator* script and result. Video 4—*Towards Adaptive Morphogenesis* in the online supplementary material shows the experiment which produced this result. NB: In this experiment, the pattern on the right is slightly malformed. This was due to the seed robot's connection slot being mis-sensed (due to camera noise) by the first robot to attach

9 Discussion

In all of our experiments, one or all of the robots were preprogrammed with a script for generating a single predefined morphology. In a real task-execution scenario, robots may need to self-assemble into several different morphologies at various stages. SWARMORPH-script allows for robots to form several different morphologies during task-execution. This could happen in one of two different ways. One possibility is to preprogram all of the robots with a homogeneous set of rules where different ID ranges are used for the different morpholo-

```
Disconnect();
Retreat(10 s);
OpenConnSlot(back);
SendScript(
   OpenConnSlot(back);
   SendScript(
      StopExecution();
   );
   StopExecution();
);
OpenConnSlot(left);
SendScript(self);
StopExecution();
```

Fig. 14 *Communicated-line-replicator* script and result. Video 4—*Towards Adaptive Morphogenesis* in the online supplementary material shows the experiment which produced this result

gies.[2] Alternatively, different robots can be preprogrammed with different morphologies and the robots could leverage their capacity to communicate whole rule sets (as demonstrated in Sect. 8).

However, SWARMORPH-script does not limit us to use prespecified morphologies. In fact, if used in combination with local, situated communication, SWARMORPH-script effectively becomes a generic morphology control platform which is agnostic with respect to different scientific approaches. A gradient or virtual hormones could be propagated instead of rule IDs. This would potentially make the morphology formation process more stochastic, thus moving towards the approaches presented in (Bojinov et al. 2000; Shen et al. 2004). Alternatively, the *s-bots* could use sensor readings to determine in which direction to open the next connection slot based on local features of the environment. For example, if the task were to encircle an object, a connected robot could use its infrared proximity sensor to determine the local shape of the object and open the next connection slot accordingly.

The current implementation of the visual communication mechanism is slow and puts a practical limit on the size of the scripts that can be communicated. Furthermore, coordinated action by physically connected robots in a morphology (such as shifting an object or overcoming an obstacle) also requires communication. An overly slow communication mechanism renders such coordination difficult for morphologies consisting of a large number of robots. However, our method would be compatible with any form of point–to–point communication. Much faster communication would be possible with slightly upgraded hardware that incorporated infrared communication or dedicated communication embedded in the physical connection mechanism. In this way, we expect that our SWARMORPH-script based approach would scale to much larger groups of robots and could potentially be generalized to other modular and self-assembling robotic systems.

[2] In (O'Grady et al. 2008) we demonstrate this approach by letting a group of robots self-assemble and subsequently reconfigure.

10 Conclusions and directions for future work

In this paper, we have demonstrated the formation of arbitrary morphologies by self-assembling robots. We defined basic capabilities that are essential to render any self-assembling morphology control system viable in a practical task-execution scenario. Any such system must: allow the formation of arbitrary morphologies, provide control over the size of a morphology, enable self-assembly into multiple separate structures, and allow robots to form morphologies of which they have no a priori knowledge. To provide our system with these capabilities, we abstracted basic control primitives as commands in a simple language, and used this high-level descriptive language to express local rules for growing morphologies. Our approach is decentralized. The robots have limited sensory capabilities and they are unable to detect the state or shape of the global structure. Robots that are not part of the morphology are only able to perceive where they should connect and what orientation they should assume. Once a robot connects to a morphology, it initiates communication with the robot to which it formed the connection in order to discover if and how to extend the morphology locally.

We showed how morphologies can be pre-specified as sets of rules and how morphologies can thus be formed by communicating rule IDs. We also showed that by communicating whole scripts instead of rule IDs, robots need not have a priori knowledge about potential morphologies in order to participate in their formation. We demonstrated how multiple morphologies can be formed both when pre-specified rules are used and when entire scripts are communicated.

Our ongoing research concerns leveraging the morphology generation approach presented in this study to add functional value to a group of robots. More specifically, we want to give the robots the capability to identify different types of obstacles, assemble into appropriate morphologies and then to overcome the obstacles. In an all-terrain navigation task, for example, the group could self-assemble into a line morphology in order to cross a ditch, while uneven or hilly terrain could trigger self-assembly into a dense morphology that provides stability.

Acknowledgements This work would not have been possible without the innovative robotic hardware developed by Francesco Mondada's group at the Laboratoire de Systèmes Robotiques of the École Polytechnique Fédérale de Lausanne. This work was supported by the *SWARMANOID* project, funded by the Future and Emerging Technologies programme (IST-FET) of the European Commission, under grant IST-022888 and by the *VIRTUAL SWARMANOID* project funded by the F.R.S.-FNRS. Anders Christensen acknowledges support from COMP2SYS, a Marie Curie Early Stage Research Training Site funded by the European Community's Sixth Framework Programme (grant MEST-CT-2004-505079). The information provided is the sole responsibility of the authors and does not reflect the European Commission's opinion. The European Commission is not responsible for any use that might be made of data appearing in this publication. Marco Dorigo acknowledges support from the F.R.S.-FNRS, of which he is a Research Director.

References

Balch, T., & Arkin, R. C. (1998). Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, *14*(6), 926–939.

Bojinov, H., Casal, A., & Hogg, T. (2000). Emergent structures in modular self-reconfigurable robots. In *Proceedings of the IEEE international conference on robotics & automation* (Vol. 2, pp. 1734–1741). Los Alamitos: IEEE Computer Society Press.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm intelligence: from natural to artificial systems*. New York: Oxford University Press.

Brown, H. B., Weghe, J. M. V., Bererton, C. A., & Khasla, P. K. (2002). Millibot trains for enhanced mobility. *IEEE/ASME Transactions on Mechatronics*, *7*(4), 452–461.

Butler, Z., Kotay, K., Rus, D., & Tomita, K. (2004). Generic decentralized control for lattice-based self-reconfigurable robots. *International Journal of Robotics Research*, *23*(9), 919–937.

Castano, A., Shen, W.-M., & Will, P. (2000). CONRO: Towards deployable robots with inter-robots metamorphic capabilities. *Autonomous Robots*, *8*(3), 309–324.

Christensen, A. L., O'Grady, R., & Dorigo, M. (2007a). A mechanism to self-assemble patterns with autonomous robots. In *Proceedings of the 9th European conference on artificial life (ECAL2007)* (pp. 716–725). Berlin: Springer.

Christensen, A. L., O'Grady, R., & Dorigo, M. (2007b). Morphology control in a multirobot system. *IEEE Robotics & Automation Magazine*, *14*(4), 18–25.

Damoto, R., Kawakami, A., & Hirose, S. (2001). Study of super-mechano colony: concept and basic experimental set-up. *Advanced Robotics*, *15*(4), 391–408.

Das, A. K., Fierro, R., Kumar, V., Ostrowski, J. P., Spletzer, J., & Taylor, C. J. (2002). A vision-based formation control framework. *IEEE Transactions on Robotics and Automation*, *18*(5), 813–825.

Dorigo, M., Trianni, V., Şahin, E., Groß, R., Labella, T. H., Baldassarre, G., Nolfi, S., Deneubourg, J.-L., Mondada, F., Floreano, D., & Gambardella, L. M. (2004). Evolving self-organizing behaviors for a swarm-bot. *Autonomous Robots*, *17*(2–3), 223–245.

Dorigo, M., Tuci, E., Trianni, V., Groß, R., Nouyan, S., Ampatzis, C., Labella, T. H., O'Grady, R., Bonani, M., & Mondada, F. (2006). SWARM-BOT: Design and implementation of colonies of self-assembling robots. In G. Y. Yen & D. B. Fogel (Eds.), *Computational intelligence: principles and practice* (pp. 103–135). New York: IEEE Computational Intelligence Society.

Fukuda, T., Buss, M., Hosokai, H., & Kawauchi, Y. (1991). Cell structured robotic system CEBOT: control, planning and communication methods. *Robotics and Autonomous Systems*, *7*(2–3), 239–248.

Groß, R., & Dorigo, M. (2008, in press). Self-assembly at the macroscopic scale. *Proceedings of the IEEE*.

Groß, R., Bonani, M., Mondada, F., & Dorigo, M. (2006). Autonomous self-assembly in swarm-bots. *IEEE Transactions on Robotics*, *22*(6), 1115–1130.

Hirose, S., Shirasu, T., & Fukushima, E. F. (1996). Proposal for cooperative robot "Gunryu" composed of autonomous segments. *Robotics and Autonomous Systems*, *17*, 107–118.

Jones, C., & Matarić, M. J. (2003). From local to global behavior in intelligent self-assembly. In *Proceedings of the 2003 IEEE international conference on robotics and automation, ICRA'03* (Vol. 1, pp. 721–726). Los Alamitos: IEEE Computer Society Press.

Kawauchi, Y., Inaba, M., & Fukuda, T. (1993). A principle of distributed decision making of cellular robotic system (CEBOT). In *Proceedings of the 1993 IEEE international conference on robotics and automation, ICRA'93* (pp. 833–838). Piscataway: IEEE Press.

Klavins, E., Ghrist, R., & Lipsky, D. (2006). A grammatical approach to self-organizing robotic systems. *IEEE Transactions on Automatic Control*, *51*(6), 949–962.

Lawton, J. R. T., Beard, R. W., & Young, B. J. (2003). A decentralized approach to formation maneuvers. *IEEE Transactions on Robotics and Automation*, *19*(6), 933–941.

Lewis, M. A., & Tan, K. H. (1997). High precision formation control of mobile robots using virtual structures. *Autonomous Robots*, *4*(4), 387–403.

Mondada, F., Pettinaro, G. C., Guignard, A., Kwee, I. V., Floreano, D., Deneubourg, J.-L., Nolfi, S., Gambardella, L. M., & Dorigo, M. (2004). SWARM-BOT: a new distributed robotic concept. *Autonomous Robots*, *17*(2–3), 193–221.

Mondada, F., Gambardella, L. M., Floreano, D., Nolfi, S., Deneubourg, J.-L., & Dorigo, M. (2005). The cooperation of swarm-bots: physical interactions in collective robotics. *IEEE Robotics & Automation Magazine*, *12*(2), 21–28.

Motomura, K., Kawakami, A., & Hirose, S. (2005). Development of arm equipped single wheel rover: effective arm-posture-based steering method. *Autonomous Robots*, *18*(2), 215–229.

Murata, S., Yoshida, E., Kamimura, A., Kurokawa, H., Tomita, K., & Kokaji, S. (2002). M-tran: Self-reconfigurable modular robotic system. *IEEE-ASME Transactions on Mechatronics*, *7*(4), 431–441.

Nouyan, S., Groß, R., Bonani, M., Mondada, F., & Dorigo, M. (2006). Group transport along a robot chain in a self-organised robot colony. In T. Arai, R. Pfeifer, T. Balch, & H. Yokoi (Eds.), *Intelligent autonomous systems* (Vol. 9, pp. 433–442). Amsterdam: IOS Press.

O'Grady, R., Groß, R., Mondada, F., Bonani, M., & Dorigo, M. (2005). Self-assembly on demand in a group of physical autonomous mobile robots navigating rough terrain. In *Lecture notes in artificial intelligence: Vol. 3630. Advances in artificial life: 8th European conference, ECAL 2005, proceedings* (pp. 272–281). Berlin: Springer.

O'Grady, R., Christensen, A. L., & Dorigo, M. (2008). Automous reconfiguration in a self-assembling multi-robot system. In *Lecture notes in computer science: Vol. 5217. Ant colony optimization and swarm intelligence, sixth international conference, ANTS 2008, proceedings* (pp. 261–268). Berlin: Springer.

Rus, D., & Vona, M. (1999). Self-reconfiguration planning with compressible unit modules. In *Proceedings of the 1999 IEEE international conference on robotics and automation, ICRA'99* (Vol. 4, pp. 2513–2520). Los Alamitos: IEEE Computer Society Press.

Rus, D., & Vona, M. (2001). Crystalline robots: self-reconfiguration with compressible unit modules. *Autonomous Robots*, *10*(1), 107–124.

Salemi, B., Moll, M., & Shen, W.-M. (2006). SUPERBOT: a deployable, multi-functional, and modular self-reconfigurable robotic system. In *Proceedings of the 2006 IEEE/RSJ international conference on intelligent robots and systems* (pp. 3636–3641). Piscataway: IEEE Press.

Shen, W.-M., Will, P., Galstyan, A., & Chuong, C. M. (2004). Hormone-inspired self-organization and distributed control of robotic swarms. *Autonomous Robots*, *17*(1), 93–105.

Shen, W.-M., Krivokon, M., Chiu, H., Everist, J., Rubenstein, M., & Venkatesh, J. (2006). Multimode locomotion for reconfigurable robots. *Autonomous Robots*, *20*(2), 165–177.

Støy, K. (2001). Using situated communication in distributed autonomous mobile robots. In *Proceedings of the 7th Scandinavian conference on artificial intelligence* (pp. 44–52). Amsterdam: IOS Press.

Støy, K. (2006). Using cellular automata and gradients to control self-reconfiguration. *Robotics and Autonomous Systems*, *54*(2), 135–141.

Støy, K., & Nagpal, R. (2004). Self-reconfiguration using directed growth. In *Proceedings of the international conference on distributed autonomous robot systems (DARS-04)* (pp. 1–10). Berlin: Springer.

Trianni, V., & Dorigo, M. (2006). Self-organisation and communication in groups of simulated and physical robots. *Biological Cybernetics*, *95*, 213–231.

White, P., Zykov, V., Bongard, J., & Lipson, H. (2005). Three dimensional stochastic reconfiguration of modular robots. In *Proceedings of robotics science and systems* (pp. 161–168). Cambridge: MIT Press.

Yamakita, M., Taniguchi, Y., & Shukuya, Y. (2003). Analysis of formation control of cooperative transportation of mother ship by SMC. In *Proceedings of the 2003 international conference on robotics and automation, ICRA'03* (Vol. 1, pp. 951–956). Los Alamitos: IEEE Computer Society Press.

Yim, M., Duff, D. G., & Roufas, K. D. (2000). PolyBot: a modular reconfigurable robot. In *Proceedings of the 2000 IEEE international conference on robotics and automation, ICRA'00* (Vol. 1). Piscataway: IEEE Press.

Yim, M., Roufas, K., Duff, D., Zhang, Y., Eldershaw, C., & Homans, S. B. (2003). Modular reconfigurable robots in space applications. *Autonomous Robots*, *14*(2–3), 225–237.

Yim, M., Shen, W.-M., Salemi, B., Rus, D., Moll, M., Lipson, H., Klavins, E., & Chirikjian, G. S. (2007). Modular self-reconfigurable robot systems. *IEEE Robotics & Automation Magazine*, *14*(1), 43–52.

Yu, C.-H., & Nagpal, R. (2008). Sensing-based shape formation on modular multi-robot systems: a theoretical study. In *Proceedings of the 7th international conference on autonomous agents and multiagent systems (AAMAS 2008)*. New York: ACM.

Yu, C.-H., Willems, F.-X., Ingber, D., & Nagpal, R. (2007). Self-organization of environmentally-adaptive shapes on a modular robot. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS 2007)* (pp. 2353–2360). Piscataway: IEEE Press.